On Using Selectional Restriction in Language Models for Speech Recognition

Joerg P. Ueberla

CMPT TR 94-03
School of Computing Science[*], Simon Fraser University,
Burnaby, B.C., V5A 1S6, CANADA
email: ueberla@cs.sfu.ca

February 5, 2008

Abstract

In this paper, we investigate the use of selectional restriction – the constraints a predicate imposes on its arguments – in a language model for speech recognition. We use an un-tagged corpus, followed by a public domain tagger and a very simple finite state machine to obtain verb-object pairs from unrestricted English text. We then measure the impact the knowledge of the verb has on the prediction of the direct object in terms of the perplexity of a cluster-based language model. The results show that even though a clustered bigram is more useful than a verb-object model, the combination of the two leads to an improvement over the clustered bigram model.

[*]This is a revised version of the original technical report, prepared for the cmp-lg server. The work was supported by NSERC operating grant OGP0041910.

1 Introduction

The constraint a predicate imposes on its arguments is called selectional restriction. For example, in the sentence fragment "She eats x", the verb "eat" imposes a constraint on the direct object "x": "x" should be something that is usually being eaten. This phenomenon has received considerable attention in the linguistic community and it was recently explained in statistical terms in [Res93]. Because it restricts subsequent nouns and because nouns account for a large part of the perplexity (see [Ueb94]), it seems natural to try to use selectional restriction in a language model for speech recognition. In this paper, we report on our work in progress on this topic. We begin by presenting in section 2 the process we use to obtain training and testing data from unrestricted English text. The data constitutes the input to a clustering algorithm and a language model, both of which are described in section 3. In section 4, we present the results we have obtained so far, followed by conclusions in section 5.

2 Training and Testing Data

In order to use selectional restriction in a language model for speech recognition, we have to be able to identify the predicate and its argument in naturally occurring, unrestricted English text in an efficient manner. Since the parsing of unrestricted text is a yet unsolved, complicated problem by itself, we do not attempt to use a sophisticated parser. Instead, we use the un-tagged version of the Wall Street Journal Corpus (distributed by ACL/DCI), Xerox's public domain tagger (described in [CKPS92]) and a very simple deterministic finite-state automaton to identify verbs and their direct objects. The resulting data is certainly very noisy, but, as opposed to more accurate data obtained from a sophisticated parser, it would be feasible to use this method in a speech recogniser. The finite-state automaton we use only has three states and it is shown in Figure 1. The circles correspond to states and the arcs to transitions. The input to the automaton consists of a sequence of words with associated tags. The words and tags [1] are classified into three different events:

- V : the word is a verb (iff its tag starts with "v", "b" or "h")
- PP: the word is a preposition (iff its tag is "in")
- NC: the word starts a new clause (iff its tag is ".",":",";","!","?","cs" or begins with "w")

All other words and tags do not lead to a change in the state of the automaton. Intuitively, state 1 corresponds to the beginning of a new clause without having seen its verb, state 2 to a clause after having seen its verb and state 3 to a prepositional

[1] "v" corresponds to most forms of most verbs, "b" corresponds to forms of "to be", "h" corresponds to forms of "to have", "cs" contains words like "although", "since" and "w" contains words like "who", "when". The tagset we use is the one provided by the tagger and for more information please refer to [CKPS92].

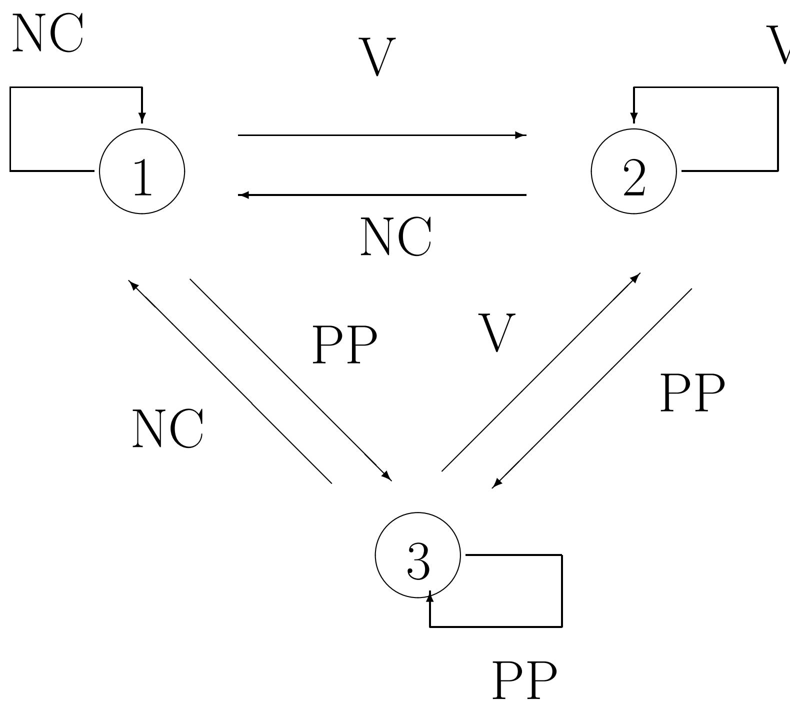

Figure 1: Finite State Automaton

phrase. Given this automaton, we then consider each occurrence of a noun (e.g. tag "nn") to be

- a direct object if we are currently in state 2; the corresponding verb is considered to be the one that caused the last transition into state 2;
- part of a prepositional phrase if we are currently in state 3; the corresponding preposition is considered to be the one that caused the last transition into state 3;
- unconstrained by a predicate or preposition (for example a subject) if we are in state 1.

The automaton then outputs a sequence of verb-object pairs, which constitute our training and testing data. Examples of the output of the automaton are shown in Table 1. Entries that would not normally be considered verb-object pairs are marked with "*". The data is very noisy because errors can be introduced both by the tagger (which makes its decisions based on a small window of words) and by the overly simplistic finite state automaton (for example, not all occurrences of "cs" and "w" constitute a new clause). Given this data, the goal of our language model is to predict the direct objects and we will measure the influence the knowledge of the preceding verb has on this prediction in terms of perplexity.

Verb	Object	
join	board	
is	chairman	*
named	director	
make	cigarette	
make	filters	
caused	percentage	
enters	lungs	
causing	symptoms	
show	decades	*
show	researchers	
Loews	Corp.	*
makes	cigarettes	
stopped	crocidolite	
bring	attention	
This	old	*
having	properties	
is	asbestos	*
studied	workers	

Table 1: Examples of the noisy verb-object data

3 The Language Model

We can formulate the task of the language model as the prediction of the value of some variable Y (e.g. the next word) based on some knowledge about the past encoded by variables $X_1, ..., X_n$. In our case, Y is the next direct object and the only knowledge we have about the past is the identity of the previous verb encoded by the variable X. The most straight forward way would be to directly estimate the conditional probability distribution $p(Y = y_l | X = x_k)$. However, because of sparse training data, it is often difficult to estimate this distribution directly. Class based models, that group elements $x_k \in \mathcal{X}$ into classes $g_x = G_1(x_k)$ and elements $y_l \in \mathcal{Y}$ into classes $g_y = G_2(y_l)$ can be used to alleviate this problem. The conditional probability distribution is then calculated as

$$p_G(y_l | x_k) = p(G_2(y_l) | G_1(x_k)) * p(y_l | G_2(y_l)), \tag{1}$$

which generally requires less training data [2]. In the following, let $(X[i], Y[i]), 1 \leq i \leq N$ denote the values of X and Y at the i^{th} data point and let $(G_1[i], G_2[i])$ denote the corresponding classes. How can we obtain the classification functions G_1 and G_2 automatically, given only a set of N data points $(X[i], Y[i])$? In the following (sections 3.1, 3.2, 3.3), we describe a method which is almost identical to the one presented in

[2]As an example, if $\mathcal{X}$ and $\mathcal{Y}$ have 10,000 elements each, and if we use 200 classes for G_1 and for G_2, then the original model has to estimate $10,000 * 10,000 = 1 * 10^8$ probabilities whereas the class based model only needs to estimate $200 * 200 + 200 * 10,000 = 2.04 * 10^6$.

[KN93]. The only difference is that in [KN93], the elements of variables X and Y are identical (the data consists of bigrams), thus requiring only one clustering function G [3]. In our case, however, the variables X and Y contain different elements (verbs and objects respectively), and we thus produce two clustering functions G_1 and G_2.

3.1 Maximum-Likelihood Criterion

In order to automatically find classification functions G_1 and G_2, which – as a shorthand – we will also denote as G, we first convert the classification problem into an optimisation problem. Suppose the function $F(G)$ indicates how good the classification G (composed of G_1 and G_2) is. We can then reformulate the classification problem as finding the classification G that maximises F:

$$G = argmax_{G' \in \mathcal{G}} F(G^{'}), \tag{2}$$

where $\mathcal{G}$ contains the set of possible classifications which are at our disposal.

What is a suitable function F, also called optimisation criterion? Given a classification function G, we can estimate the probabilities $p_G(y_l|x_k)$ of equation 1 using the maximum likelihood estimator, e.g. relative frequencies:

$$\begin{aligned} p_G(y_l|x_k) &= p(G_2(y_l)|G_1(x_k)) * p(y_l|G_2(y_l)) && (3) \\ &= \frac{N(g_x, g_y)}{N(g_x)} * \frac{N(g_y, y)}{N(g_y)}, && (4) \end{aligned}$$

where $g_x = G_1(x_k)$, $g_y = G_2(y_l)$ and $N(x)$ denotes the number of times x occurs in the data. Given these probability estimates $p_G(y_l|x_k)$, the likelihood F_{ML} of the training data, e.g. the probability of the training data being generated by our probability estimates $p_G(y_l|x_k)$, measures how well the training data is represented by the estimates and can be used as optimisation criterion ([Jel90]).

In the following, we will derive an optimisation function F_{ML} in terms of frequency counts observed in the training data. The likelihood of the training data F_{ML} is simply

$$\begin{aligned} F_{ML} &= \prod_{i=1}^{N} p_G(Y[i]|X[i]) && (5) \\ &= \prod_{i=1}^{N} \frac{N(G_1[i], G_2[i])}{N(G_1[i])} * \frac{N(G_2[i], Y[i])}{N(G_2[i])}. && (6) \end{aligned}$$

Assuming that the classification is unique, e.g. that G_1 and G_2 are functions, we have $N(G_2[i], Y[i]) = N(Y[i])$ (because $Y[i]$ always occurs with the same class $G_2[i]$). Since we try to optimise F_{ML} with respect to G, we can remove any term that does not depend on G, because it will not influence our optimisation. It is thus equivalent

[3] It is possible that using two clustering functions would be beneficial even if the two variables have the same elements.

to optimise

$$F'_{ML} = \prod_{i=1}^{N} f(X[i], Y[i]) \tag{7}$$

$$= \prod_{i=1}^{N} \frac{N(G_1[i], G_2[i])}{N(G_1[i])} * \frac{1}{N(G_2[i])}. \tag{8}$$

If, for two pairs $(X[i], Y[i])$ and $(X[j], Y[j])$, we have $G_1(X[i]) = G_1(X[j])$ and $G_2(Y[i]) = G_2(Y[j])$, then we know that $f(X[i], Y[i]) = f(X[j], Y[j])$. We can thus regroup identical terms to obtain

$$F'_{ML} = \prod_{g_x, g_y} [\frac{N(g_x, g_y)}{N(g_x)} * \frac{1}{N(g_y)}]^{N(g_x, g_y)}, \tag{9}$$

where the product is over all possible pairs (g_x, g_y). Because $N(g_x)$ does not depend on g_y and $N(g_y)$ does not depend on g_x, we can simplify this again to

$$F'_{ML} = \prod_{g_x, g_y} N(g_x, g_y)^{N(g_x, g_y)} \prod_{g_x} \frac{1}{N(g_x)}^{N(g_x)} \prod_{g_y} \frac{1}{N(g_y)}^{N(g_y)}. \tag{10}$$

Taking the logarithm, we obtain the equivalent optimisation criterion

$$F''_{ML} = \sum_{g_x, g_y} N(g_x, g_y) * log(N(g_x, g_y)) - \sum_{g_x} N(g_x) * log(N(g_x)) \tag{11}$$

$$- \sum_{g_y} N(g_y) * log(N(g_y)).$$

F_{ML} is the maximum likelihood optimisation criterion and we could use it to find good classifications G. However, the problem with this maximum likelihood criterion is that we first estimate the probabilities $p_G(y_l|x_k)$ on the training data T and then, given $p_G(y_l|x_k)$, we evaluate the classification G on T. In other words, both the classification G and the estimator $p_G(y_l|x_k)$ are trained on the same data. Thus, there will not be any unseen event, a fact that overestimates the power for generalisation of the class based model. In order to avoid this, we will in section 3.2 incorporate a cross-validation technique directly into the optimisation criterion.

3.2 Leaving-One-Out Criterion

The basic principle of cross-validation is to split the training data T into a "retained" part T_R and a "held-out" part T_H. We can then use T_R to estimate the probabilities $p_G(y_l|x_k)$ for a given classification G, and T_H to evaluate how well the classification G performs. The so-called leaving-one-out technique is a special case of cross-validation ([DH73, pp.75]). It divides the data into $N - 1$ samples as "retained" part and only one sample as "held-out" part. This is repeated $N - 1$ times, so that each sample is once in the "held-out" part. The advantage of this approach is that all samples are used in the "retained" and in the "held-out" part, thus making very efficient use of the existing data. In other words, our "held-out" part T_H to evaluate a classification G is the entire set of data points; but when we calculate the probability of the i^{th} data point, we assume that our probability distribution $p_G(y_l|x_k)$ was estimated on all the data expect point i.

Let T_i denote the data without the pair $(X[i], Y[i])$ and $p_{G,T_i}(y_l|x_k)$ the probability estimates based on a given classification G and training corpus T_i. Given a particular T_i, the probability of the "held-out" part $(X[i], Y[i])$ is $p_{G,T_i}(y_l|x_k)$. The probability of the complete corpus, where each pair is in turn considered the "held-out" part is the leaving-one-out likelihood L_{LO}

$$L_{LO} = \prod_{i=1}^{N} p_{G,T_i}(Y[i]|X[i]). \tag{12}$$

In the following, we will derive an optimisation function F_{LO} by specifying how $p_{G,T_i}(Y[i]|X[i])$ is estimated from frequency counts. First we rewrite $p_{G,T_i}(Y[i]|X[i])$ as usual (see equation 1):

$$\begin{aligned} p_{G,T_i}(y_l|x_k) &= P_{G,T_i}(G_2(y_l)|G_1(x_k)) * P_{G,T_i}(y_l|G_2(y_l)) && (13) \\ &= \frac{p_{G,T_i}(g_x, g_y)}{p_{G,T_i}(g_x)} * \frac{p_{G,T_i}(g_y, y_l)}{p_{G,T_i}(g_y)}, && (14) \end{aligned}$$

where $g_x = G_1(x_k)$ and $g_y = G_2(y_l)$. Now we will specify how we estimate each term in equation 14.

As we saw before, $p_{G,T_i}(g_y, y_l) = p_{G,T_i}(y_l)$ (if the classification G_2 is a function) and since $p_{T_i}(y_l)$ is actually independent of G, we can drop it out of the maximization and thus need not specify an estimate for it.

As we will see later, we can guarantee that every class g_x and g_y has been seen at least once in the "retained" part and we can thus use relative counts as estimates for class uni-grams:

$$\begin{aligned} p_{G,T_i}(g_x) &= \frac{N_{T_i}(g_x)}{N_{T_i}} && (15) \\ p_{G,T_i}(g_y) &= \frac{N_{T_i}(g_y)}{N_{T_i}}. && (16) \end{aligned}$$

In the case of the class bi-gram, we might have to predict unseen events [4]. We therefore use the absolute discounting method ([NE93]), where the counts are reduced by a constant value $b < 1$ and where the gained probability mass is redistributed over unseen events. Let n_{0,T_i} be the number of unseen pairs (g_x, g_y) and n_{+,T_i} the number of seen pairs (g_x, g_y), leading to the following smoothed estimate

$$\begin{aligned} & p_{G,T_i}(g_x, g_y) \\ &= \begin{cases} \frac{N_{T_i}(g_x,g_y)-b}{N_{T_i}} & \text{if } N_{T_i}(g_x, g_y) > 0 \\ \frac{n_{+,T_i}*b}{n_{0,T_i}*N_{T_i}} & \text{if } N_{T_i}(g_x, g_y) = 0 \end{cases} && (17) \end{aligned}$$

[4]If $(X[i], Y[i])$ occurs only once in the complete corpus, then $p_{G,T_i}(Y[i]|X[i])$ will have to be calculated based on the corpus T_i, which does not contain any occurrences of $(X[i], Y[i])$.

Ideally, we would make b depend on the classification, e.g. use $b = \frac{n_1}{n_1+2*n_2}$, where n_1 and n_2 depend on G. However, due to computational reasons, we use, as suggested in [KN93], the empirically determined constant value $b = 0.75$ during clustering. The probability distribution $p_{G,T_i}(g_x, g_y)$ will always be evaluated on the "held-out" part $(X[i], Y[i])$ and with $g_{x,i} = G_1[i]$ and $g_{y,i} = G_2[i]$ we obtain

$$\begin{aligned} & p_{G,T_i}(g_{x,i}, g_{y,i}) \\ &= \begin{cases} \frac{N_{T_i}(g_{x,i},g_{y,i})-b}{N_{T_i}} & \text{if } N_{T_i}(g_{x,i}, g_{y,i}) > 0 \\ \frac{n_{+,T_i}*b}{n_{0,T_i}*N_{T_i}} & \text{if } N_{T_i}(g_{x,i}, g_{y,i}) = 0 \end{cases} \end{aligned} \tag{18}$$

In order to facilitate future regrouping of terms, we can now express the counts $N_{T_i}, N_{T_i}(g_x)$ etc. in terms of the counts of the complete corpus T as follows:

$$N_{T_i} = N_T - 1 \tag{19}$$
$$N_{T_i}(g_x) = N_T(g_x) - 1 \tag{20}$$
$$N_{T_i}(g_y) = N_T(g_y) - 1 \tag{21}$$
$$N_{T_i}(g_{x,i}, g_{y,i}) = N_T(g_{x,i}, g_{y,i}) - 1 \tag{22}$$
$$N_{T_i} = N_T - 1 \tag{23}$$
$$n_{+,T_i} = \begin{cases} n_{+,T} & \text{if } N_T(g_{x,i}, g_{y,i}) > 1 \\ n_{+,T} - 1 & \text{if } N_T(g_{x,i}, g_{y,i}) = 1 \end{cases} \tag{24}$$
$$n_{0,T_i} = \begin{cases} n_{0,T} & \text{if } N_T(g_{x,i}, g_{y,i}) > 1 \\ n_{0,T} - 1 & \text{if } N_T(g_{x,i}, g_{y,i}) = 1 \end{cases} \tag{25}$$

All we have left to do now is to substitute all the expressions back into equation 12. After dropping $p_{G,T_i}(y_l)$ because it is independent of G we get

$$F'_{LO} = \prod_{i=1}^{N} \frac{p_{G,T_i}(g_{x,i}, g_{y,i})}{p_{G,T_i}(g_{x,i})} * \frac{1}{p_{G,T_i}(g_{y,i})} \tag{26}$$
$$= \prod_{g_x,g_y} (p_{G,T_i}(g_x, g_y))^{N(g_x,g_y)} * \prod_{g_x} (\frac{1}{p_{G,T_i}(g_x)})^{N(g_x)} * \prod_{g_y} (\frac{1}{p_{G,T_i}(g_y)})^{N(g_y)}. \tag{27}$$

We can now substitute equations 15, 16 and 18, using the counts of the whole corpus of equations 19 to 25 . After having dropped terms independent of G, we obtain

$$\begin{aligned} F''_{LO} &= \prod_{g_x,g_y:N(g_x,g_y)>1} (N_T(g_x, g_y) - 1 - b)^{N_T(g_x,g_y)} * \left(\frac{(n_{+,T} - 1) * b}{(n_{0,T} + 1)}\right)^{n_{1,T}} \\ &* \prod_{g_x} \left(\frac{1}{(N_T(g_x - 1))}\right)^{N_T(g_x)} * \prod_{g_y} \left(\frac{1}{(N_T(g_y - 1))}\right)^{N_T(g_y)}, \end{aligned} \tag{28}$$

where $n_{1,T}$ is the number of pairs (g_x, g_y) seen exactly once in T (e.g. the number of pairs that will be unseen when used as "held-out" part). Taking the logarithm, we obtain the final optimisation criterion F'''_{LO}

$$F'''_{LO} = \sum_{g_x,g_y:N_T(g_x,g_y)>1} N_T(g_x, g_y) * log(N_T(g_x, g_y) - 1 - b) \tag{29}$$

```
-start with an initial clustering function G
-iterate until some convergence criterion is met
{
     -for all x in  X and y in Y
     {
          -for all gx and gy
          {
               -calculate the difference in F(G) when
               x/y is moved from its current cluster to gx/gy
          }
          -move the x/y that results in the biggest improvement
          in the value of F(G)
     }
}
```

Figure 2: The clustering algorithm

$$
\begin{aligned}
&+ \quad n_{1,T} * log(\frac{b * (n_{+,T} - 1)}{(n_{0,T} + 1)}) \\
&- \quad \sum_{g_x} N_T(g_x) * log(N_T(g_x) - 1) - \sum_{g_y} N_T(g_y) * log(N_T(g_y) - 1).
\end{aligned}
$$

3.3 Clustering Algorithm

Given the F'''_{LO} maximization criterion, we use the algorithm in Figure 2 to find a good clustering function G. The algorithm tries to make local changes by moving words between classes, but only if it improves the value of the optimisation function. The algorithm will converge because the optimisation criterion is made up of logarithms of probabilities and thus has an upper limit and because the value of the optimisation criterion increases in each iteration. However, the solution found by this greedy algorithm is only locally optimal and it depends on the starting conditions. Furthermore, since the clustering of one word affects the future clustering of other words, the order in which words are moved is important. As suggested in [KN93], we sort the words by the number of times they occur such that the most frequent words, about which we know the most, are clustered first. Moreover, we do not consider infrequent words (e.g. words with occurrence counts smaller than 5) for clustering, because the information they provide is not very reliable. Thus, if we start out with an initial clustering in which no cluster occurs only once, and if we never move words that occur only once, then we will never have a cluster which occurs only once. Thus, the assumption we made earlier, when we decided to estimate cluster uni-grams by frequency counts, can be guaranteed.

We will now determine the complexity of the algorithm. Let M_X and M_Y be the maximal number of clusters for X and Y, let $|X|$ and $|Y|$ be the number of possible values for X and Y, let $M = max(M_X, M_Y)$, $W = max(|X|, |Y|)$ and let I be the number of iterations. When we move x from g_x to g'_x in the inner loop (the situation is symmetrical for y), we need to change the counts $N(g_x, g_y)$ and $N(g'_x, g_y)$ for all g_y. The amount by which we need to change the counts is equal to the number of times X occurred with cluster g_y. Since this amount is independent of g'_x, we need to calculate it only once for each x. The amount can then be looked up in constant time within the loop, thus making the inner loop of order M. The inner loop is executed once for every cluster x can be moved to, thus giving a complexity of the order of M^2. For each x, we needed to calculate the number of times x occurred with all clusters g_y. For that we have to sum up all the bigram counts $N(x, y) : G_2(y) = g_y$, which is on the order of W, thus giving a complexity of the order of $W + M^2$. The two outer loops are executed I and W times thus giving a total complexity of the order of $I * W * (W + M^2)$.

4 Results

In the experiments performed so far, we work with $200,000$ verb-object pairs extracted from the 1989 section of the Wall Street Journal corpus. The data contains about $19,000$ different direct object tokens, about $10,000$ different verb tokens and about $140,000$ different token pairs. We use $\frac{3}{4}$ of the data as training and the rest as testing data. For computational reasons, we have so far restricted the number of possible clusters to 50, but we hope to be able to increase that in the future. The perplexity on the testing text using the clustering algorithm on the verb-object pairs is shown in Table 2. For comparison, the table also contains the perplexity of a normal uni-gram model (e.g. no predictor variable X) and the performance of the clustering algorithm on the usual bi-gram data (e.g. the word immediately preceding the direct object as predictor variable X). We can see that the verb contains considerable information about the direct object and leads to a reduction in perplexity of about 18%. However, the immediately preceding word leads to an even bigger reduction of about 34%. We also tried a linear interpolation of the two clustered models

$$p_{interpol}(y_l) = \lambda * p_{verb-object}(y_l) + (1 - \lambda) * p_{bigram}(y_k). \tag{30}$$

On a small set of unseen data, we determined the best value (out of 50 possible values in $]0, 1[$) of the interpolation parameter λ. As shown in Table 2, the interpolated model leads to an overall perplexity reduction of 43% compared to the uni-gram, which corresponds to a reduction of about 10% over the normal bi-gram perplexity.

5 Conclusions

From a purely linguistic perspective, it would be slightly surprising to find out that the word immediately preceding a direct object can be used better to predict it than the preceding verb. However, this conclusion can not be drawn from our results because of the noisy nature of the data. In other words, the data contains pairs like

Model	Perplexity	% Reduction
uni-gram	3200	not appl.
clustered verb-object	2640	18%
clustered bigram	2010	37%
combined model	1820	43%

Table 2: Perplexity results

(is, chairman), which would usually not be considered as a verb-direct object pair. It is possible, that more accurate data (e.g. fewer, but only correct pairs) would lead to a different result. But the problem with fewer pairs would of course be that the model can be used in fewer cases, thus reducing the usefulness to a language model that would predict the entire text (rather than just the direct objects). The results thus support the common language modeling practice, in that bi-gram events (by themselves) seem to be more useful than this linguistically derived predictor (by itself). Nevertheless, the interpolation results also show that this linguistically derived predictor is useful as a complement to a standard class based bigram model. In the future, we hope to consolidate these early findings by more experiments involving a higher number of clusters and a larger data set.

References

[CKPS92] Doug Cutting, Julian Kupiec, Jan Perdersen, and Penelope Sibun. A practical part-of-speech tagger. In *Conference on Applied Natural Language Processing*, pages 133–140. Trento, Italy, 1992.

[DH73] R. O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis.* Wiley, New York, 1973.

[Jel90] Fred Jelinek. Self-organized language modeling for speech recognition. In Alex Waibel and Kai-Fu Lee, editors, *Readings in Speech Recognition*, pages 450–506. Morgan Kaufmann, San Mateo, CA, 1990.

[KN93] Reinhard Kneser and Hermann Ney. Improved clustering techniques for class-based statistical language modelling. In *European Conference on Speech Communication and Technology*, pages 973–976. Berlin, Germany, September 1993.

[NE93] Hermann Ney and Ute Essen. Estimating 'small' probabilities by leaving-one-out. In *European Conference on Speech Communication and Technology*, pages 2239–2242. Berlin, Germany, 1993.

[Res93] Philip Stuart Resnik. *Selection and Information: A Class-Based Approach to Lexical Relationships.* Ph.D. Thesis, Computer and Information Science, University of Pennsylvania, PA, 1993.

[Ueb94] Joerg Peter Ueberla. *Analyzing and Improving Statistical Language Models for Speech Recognition.* PhD thesis, School of Computing Science, Simon Fraser University, Burnaby, B.C., V5A 1S6, Canada, May 1994.